Cracking the Enigma Code: Beyond the Bombe

Adam S. Downs
asdowns114@gmail.com
Department of Mathematics and Statistics
Radford University
Radford, Virginia, USA

Neil P. Sigmon
npsigmon@radford.edu
Department of Mathematics and Statistics
Radford University
Radford, Virginia, USA

Richard E. Klima
klimare@appstate.edu
Department of Mathematical Sciences
Appalachian State University
Boone, North Carolina, USA

Abstract: *The work of Allied codebreakers in the cryptanalysis of the Enigma cipher machine during World War II has been well-documented, and rightfully recognized as one of the supreme achievements of the human intellect. One of the trickiest parts of this analysis related to the plugboard, which contributed by far the largest factor to the total number of configurations of the entire machine. To determine the daily plugboard connections, codebreakers used electromechanical devices called the bombe and checking machine. After they found the plugboard connections though, they still needed to discover some additional machine settings. The difficulty in discovering these additional settings varied depending on which branch of the German military had created the messages under attack, since each branch used slightly different procedures. The process was in fact significantly more difficult for messages created by the German Navy, as opposed to those created by either the German Army or Air Force. In this paper, we will describe and demonstrate some of the procedures involved in recovering this additional information that was needed to fully cryptanalyze the Enigma machine. To assist in demonstrating these procedures, technology involving Maplets will be used.*

1 Introduction

In 1918, German electrical engineer Arthur Scherbius applied for a patent for a mechanical cipher machine. This machine, later marketed commercially under the name *Enigma*, was designed with electric current passing through revolving wired wheels. Scherbius offered the machine to the German

military. Only after learning that their World War I ciphers had routinely been broken did they adopt it, which they used as their primary resource for encrypted communications throughout World War II.

As we described in [3], Polish codebreakers led by Marian Rejewski successfully cracked the Germans' first implementation of the Enigma. After modifications to the machine by German cryptographers rendered the Polish techniques no longer viable, British mathematician Alan Turing identified weaknesses in the encryption process using patterns generated by *cribs*, which are small parts of plaintext corresponding to known parts of the corresponding ciphertext, and which were made easier to find through the frequent mistaken use of salutations, titles, and addresses by operators. When an electromechanical device designed by Turing called the *bombe* became operational, codebreakers began again reading German messages. For messages created by the German Army and Air Force, once one encrypted message between two specific operators had been broken, every message between any two operators on the same day could be broken due to a deficiency in how Enigma operators transmitted their rotor starting positions. However, German Navy operators used a more secure procedure for transmitting their positions, which made the codebreaking process much more difficult.

In this paper, we will examine some of the challenges and illustrate some of the techniques used by Allied codebreakers to discover daily Enigma machine settings and break Enigma messages sent by operators from all branches of the German military. We will begin with a brief review of the components of an Enigma and the basics of how the bombe was used in the initial cryptanalysis.

2 Components of Enigma and Initial Cryptanalysis

We begin by reviewing the components of an Enigma machine and the challenges it presented to Allied codebreakers. We give more detailed descriptions of these components and challenges in [2]. Figure 1 shows an image of an Enigma with several important components labeled.

Figure 1: Enigma cipher machine.

To encrypt or decrypt a letter using an Enigma, the key labeled with the letter on the keyboard was pressed, which launched electric current representing the letter into the machine. The current traveled first through the plugboard, where it could change to another letter depending on whether sockets labeled with the letter were connected by a cable to sockets labeled with the other letter. The current then passed through three rotors, in each of which it could change to another letter, and then through a reflector, where it definitely changed to another letter. After this the current went back through the rotors in the opposite direction and then back through the plugboard, in each of which it could again change to another letter. Finally, the current went to the lampboard, where it lit a lamp labeled with the encrypted or decrypted letter. We describe this process in much more detail in [4].

For the plugboard, the Germans settled on using 10 cables to connect 20 letters in pairs. Using a formula that we describe and justify in [2], we find that there are 150,738,274,937,250 different ways to choose 20 letters and connect these letters in pairs using 10 cables at the plugboard.

Around each of the three rotors in an Enigma, a ring was etched with the 26 letters (or numbers representing the letters) alphabetically clockwise when viewed from the right. Above each rotor slot, a small window was cut to make the letter (or number) at a particular location on the ring visible, called the *window letter*. While each rotor could be rotated into any of 26 different positions inside the machine, just the ring could also be rotated while the central part of the rotor was held fixed. A rotation of the entire rotor versus just the ring was distinguished using a number called the *ring setting*. The ring setting essentially indicates a rotation of just the ring, while the window letter indicates a combined rotation of the central part of the rotor with the ring attached. The window letter and the ring setting each contribute a factor of $26^3 = 17{,}576$ to the total number of machine configurations.

The window letter and the ring setting must be considered separately to account for all of the variability in the rotors because the rotors rotated in the machine during encryption and decryption, rotation which was only influenced by the ring, not the central part of the rotor. Encryption and decryption was done one letter at a time, and each time an input letter was pressed on the keyboard, the rightmost rotor (before the current reached it) would rotate one position counterclockwise when viewed from the right. Notches cut into the ring around each rotor then influenced the rotation of the rotor to its left. The German Army and Air Force used rotors labeled **I–V**, into which one notch was cut into the ring. The German Navy also used additional rotors labeled **VI–VIII**, into which two notches were cut into the ring. Since each notch was on the ring, its position in the rotor slot at any time could be identified by the window letter. For each notch, there was one particular position in the rotor slot, identified by a window letter called the *notch letter*, for which if the rotor rotated one position counterclockwise, the notch would cause a rotor to its left to also rotate one position counterclockwise. Table 1 shows the notch letters for each of the rotors **I–VIII**.

Rotor	**Notch letters**	**Rotor to left rotates when window letter changes as indicated**
I	Q	Q $\rightarrow$ R
II	E	E $\rightarrow$ F
III	V	V $\rightarrow$ W
IV	J	J $\rightarrow$ K
V	Z	Z $\rightarrow$ A
VI	M and Z	M $\rightarrow$ N or Z $\rightarrow$ A
VII	M and Z	M $\rightarrow$ N or Z $\rightarrow$ A
VIII	M and Z	M $\rightarrow$ N or Z $\rightarrow$ A

Table 1: Notch letters for Enigma rotors **I–VIII**.

Since the German Army and Air Force chose from among five rotors for the three rotor slots, there were $5 \cdot 4 \cdot 3 = 60$ ways for them to arrange rotors in the machine. Since the German Navy chose from among eight rotors though, there were $8 \cdot 7 \cdot 6 = 336$ ways for them to arrange rotors in the machine. Although rotors could only be situated with each side facing in a particular direction, current passed through them in both directions, from right to left before the reflector, and then from left to right after the reflector. Like the rotors, reflectors had electrical contacts representing the 26 letters alphabetically clockwise when viewed from the right, though only on one side, since current entered and exited reflectors on the same side. Unlike the rotors (and the plugboard), letters in reflectors were always fully connected in 13 pairs. Having the reflector in the middle of the symmetric sequence

$$\text{plugboard} \rightarrow \text{rotors (right to left)} \rightarrow \text{reflector} \rightarrow \text{rotors (left to right)} \rightarrow \text{plugboard}$$

also ensured that the machine configuration for encrypting and decrypting the same message was identical, since the decryption of a ciphertext letter was obtained by simply reversing the path through the machine when the corresponding plaintext letter had been encrypted.

Reflectors labeled B and C with two different electrical wirings were produced for the German Army, Air Force, and Navy. The reflector in use was always placed in the machine in only one way and did not rotate. As such, the number of ways in which a reflector could be chosen and placed into the machine was 2.

Combining the factors resulting from the plugboard, window letters, ring settings, rotor arrangements, and reflector, the total number of different machine configurations for an Enigma used by the German Army and Air Force was 5.59×10^{24}. In an Enigma used by the German Navy, for which the number of rotor arrangements was 336 rather than 60, the total number of different machine configurations was 3.13×10^{25}.

Allied codebreakers discovered that identifying cribs allowed them to eliminate the largest factor in the total number of different machine configurations, the one resulting from the plugboard. Through a geometric pairing of the crib and its corresponding part of the ciphertext known as a *menu*, a search was done for the reflector, rotor order, window letters, and ring settings that gave plugboard pairs that produced logical consistencies, in which each letter in the menu always had a unique plugboard partner assignment.

During the search for the machine settings that gave the plugboard pairs though, the window letters and ring settings that were found were usually not the ones that had been used in the actual encryption of the message. Instead, the window letters and ring settings that were found resulted in the same rotor core starting positions, or *rotor offsets*, that had been used in the actual encryption of the message. To understand this idea better, consider an encryption with initial left-to-right window letters NCS (which correspond to position numbers 14, 3, 19 in the alphabet) and corresponding ring settings 4, 26, 13. The rotor offsets for this encryption would then be 10, 3, 6, since $14-4=10$, $3-26=-23=3 \bmod 26$, and $19-13=6$. However, if we did the same encryption with these rotors and initial window letters ZZZ (corresponding to 26, 26, 26) and corresponding ring settings 16, 23, 20, the rotor offsets would again be 10, 3, 6, since $26-16=10$, $26-23=3$, and $26-20=6$. Then, assuming the same reflector and rotor order were used, and that there were no rotations of the middle and left rotors in either scenario, the encryption using these rotors with the initial window letters NCS and corresponding ring settings 4, 26, 13 would be the same as with the initial window letters ZZZ and corresponding ring settings 16, 23, 20. To test for the correct rotor core starting positions, a setting for the initial window letters, in many cases ZZZ, was chosen, and the ring settings were found by a brute force attack from among $26^3 = 17{,}576$ possibilities, until the correct rotor core starting positions were

found. To find a plugboard partner for the central letter that was logically consistent, a combination of a reflector (either B or C), a rotor order (of which there were 60 possibilities for Army and Air Force messages or 336 possibilities for Navy messages), and a ring setting (of which there were 17,576 possibilities) was tested, which eliminated having to test the $26^3 = 17{,}576$ possibilities for the window letters.

This process allowed Allied codebreakers to break a single encrypted message between two individual Enigma operators on any particular day. To break all of the messages sent by any Enigma operators on that same day though, more work was required. We describe this next.

3 Army and Air Force Encryption

German Army and Air Force Enigma operators were equipped with a three-rotor Enigma machine and a set of five different rotors labeled **I**–**V**. Each operator was also issued the same monthly setting sheet. For each day of the month, the setting sheet would identify to the operator the machine setting, or key, for the day. Included in the daily key were the specific rotors to be installed in the machine, the prescribed order for these rotors, the choice of reflector between B and C, the ring settings for the rotors, and the ten plugboard letter pairs.

When an Enigma operator needed to encrypt a message, they set up the machine using the specified daily common key. However, when placing a rotor in the machine, each individual Enigma operator would choose their own window letter for it. The three window letters they chose for all three rotors, in order from left to right, were known as the *message setting*.

Of course, the Enigma operator to whom they were sending the message needed to know the message setting as well. We will describe a procedure for providing this that was used by German Army and Air Force Enigma operators after May 1940. For the purpose of communicating their message setting, the sending operator would choose another three letter group, known as the *indicator setting*, and start by setting up their machine using the indicator setting rather than the message setting. The sending operator would then use this setup to encrypt the message setting. Then the sending operator would change the window letters to match the message setting, and encrypt the message. The result was then transmitted to the receiving operator by sending the encrypted plaintext message following by the six letter sequence consisting of the sending operator's indicator setting and encrypted message setting.

We will demonstrate this process using a Maplet[1] entitled **Encrypt/Decrypt Enigma Message** which was written by the authors and designed to encrypt and decrypt messages using an Enigma.[2] The source code for this and all of the other Maplets demonstrated in this paper, as well as directly usable versions of them, can be downloaded at [5]. The code is unique to Maple, but could easily be altered for use with any programming language.

Suppose, for example, that the daily Enigma key is the following.

Rotor Order:	**I**, **V**, **III**
Reflector:	B
Ring Settings:	19, 13, 25
Plugboard Pairs:	O/S, N/T, R/H, B/U, L/I, D/F, G/A, Q/P, C/K, J/Y

[1] A Maplet is like an applet, but uses (and requires) the engine of the computer algebra system Maple, and is written using Maple functions and syntax.

[2] The Enigma machine used a typewriter style in which one letter was encrypted or decrypted at a time. This Maplet is designed to expedite this process by encrypting or decrypting all letters with one click.

Suppose also that the sending operator chooses message setting TEX, and indicator setting RED. The sending operator would first set up the machine using the daily key with the indicator setting as the window letters, and use this setup to encrypt the message setting. Figure 2 shows the result of using the **Encrypt/Decrypt Enigma Message** Maplet to do this, resulting in the output `YXS`.

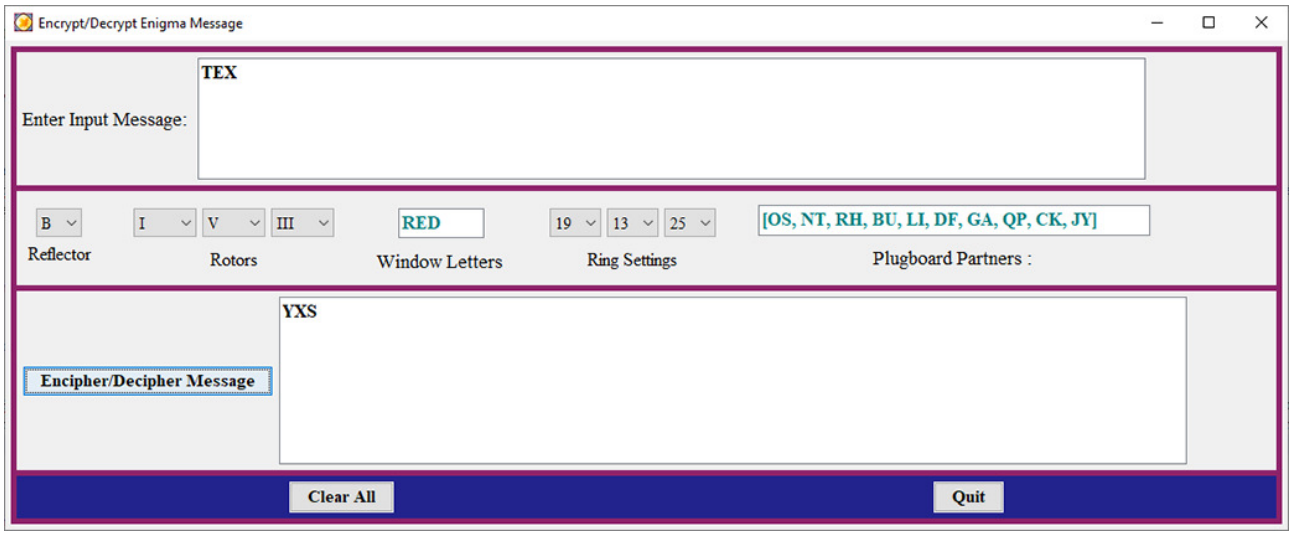


Figure 2: Encrypting a message setting.

Suppose now that the sending operator wished to encrypt the message `TODAY IS THE DAY`. The sending operator would then change the window letters in the machine to TEX, and use this setup to encrypt the message. Figure 3 shows the result of using the Maplet to do this, resulting in the ciphertext `HKMQWALFVHABJ`.

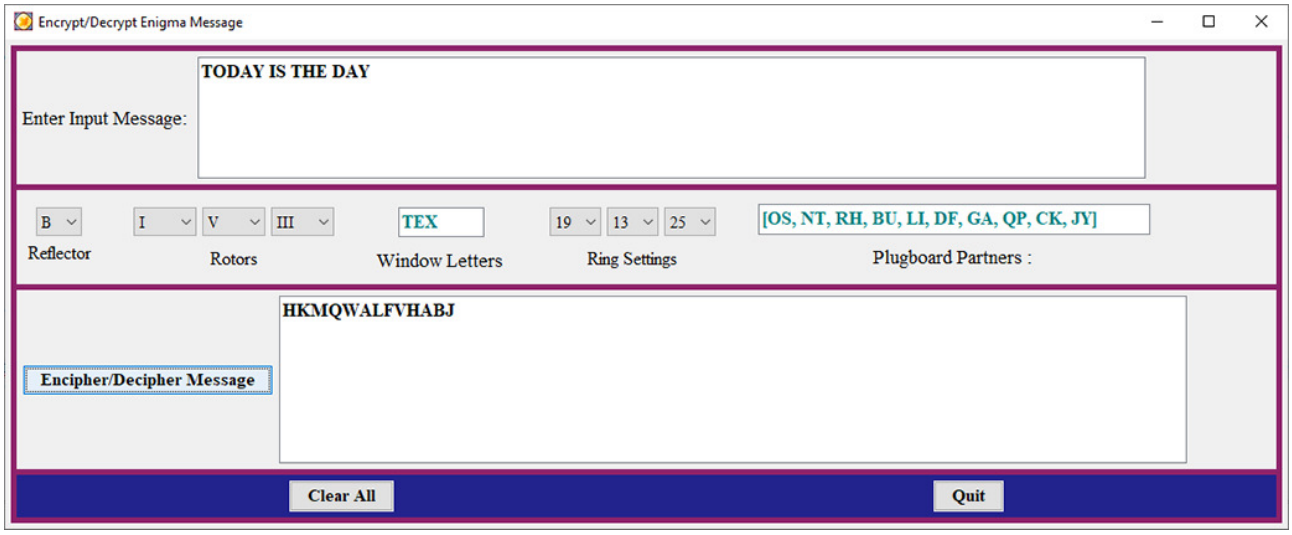


Figure 3: Encrypting a message.

Finally, the sending operator would transmit the ciphertext followed by the indicator setting and the encrypted message setting: `HKMQWALFVHABJ REDYXS`. To decrypt the message, the receiving operator would first set up their machine using the daily key with the indicator setting `RED` as the window letters, and decrypt the last three letters received. This would result in `TEX`. The receiving

operator would then change the window letters in the machine to TEX, and use this setup to decrypt the message.

4 Army and Air Force Cryptanalysis

Turing's bombe enabled Allied codebreakers to decrypt messages that had been encrypted by German Army and Air Force Enigma operators. However, there was not enough time to use the bombe to decrypt all messages sent between any two Army or Air Force operators. We will now consider how it was possible for Allied codebreakers to quickly decrypt all messages sent between any two Army or Air Force operators on the same day as one message they had decrypted using the bombe.

When an individual Enigma ciphertext was broken using the bombe, the process resulted in ring settings that gave rotor core starting positions based on the assumption that the last window letters prior to the crib were ZZZ. When the bombe was successful on one Enigma ciphertext, although it would give the reflector, rotor order, and plugboard connections from the daily key on the day the ciphertext was formed, it would only give the ring settings from the daily key if the last window letters prior to the crib had actually been ZZZ. Finding the actual ring settings from the daily key was necessary in order to be able to decrypt most or all of the other encrypted messages sent between any two Army or Air Force operators on the same day without having to use the bombe.

The first step in finding the ring settings from the daily key was to determine the actual initial window letter and ring setting of the rightmost rotor. These could be found from knowing where a turnover of the middle rotor first occurred, the notch letter of the rightmost rotor, and the rightmost initial window letter resulting from the assumption that the last window letters prior to crib were ZZZ.

To demonstrate this, we will continue an example that we began in [2], in which we assumed we knew the crib FOLLOWORDERSTO corresponded to the underlined part of the ciphertext message NUENTZERLOHHBTDSHLHIYWEABHTQKC. Using this fact, we were able to determine in [2] the following parameters from using the bombe on this one ciphertext.

Rotor Order:	**I**, **V**, **III**
Reflector:	C
Ring Settings:	16, 23, 18
Plugboard Pairs:	O/S, N/T, R/H, B/U, L/I, D/F, G/A, Q/P, C/K, J/Y

We will now demonstrate how the actual initial window letter and ring setting of the rightmost rotor can be determined for this example. We will do this using a Maplet entitled **Clonking**, which was written by the authors and designed for this purpose. The result is given in Figure 4. In the Maplet, the reflector, ring settings, rotor order, and plugboard pairs discovered using the bombe are entered, as well as the crib, its corresponding part of the ciphertext, and the ciphertext in full. Clicking the **Display Plaintext/Ciphertext** button then causes the window letters, plaintext, and ciphertext to be displayed under the assumption that the last window letters prior to the crib were ZZZ. From the output shown in the Maplet, we can see that the decrypted message is legible only through the window letters ZZS. The reason the message is no longer legible after this point is very likely due to a turnover of the middle rotor at this point. Since there are two letters in the ciphertext before the start of the crib, the Maplet thus indicates that the actual encryption starts with the window letters ZZX. So to find the actual window letter of the rightmost rotor when the ciphertext was formed, note that since a turnover of the middle rotor first occurred at ZZS and the machine was set to ZZX, then a total of 21 turnovers occurred for the rightmost rotor during encryption. From Table 1, we can see that

the rightmost rotor **III** only has the notch letter V. After 21 turnovers of the rightmost rotor during encryption, the notch letter V would have been the rightmost window letter. Thus, if we go backwards on the ring of letters around the rotor (i.e., backwards in the alphabet, wrapping from the start of the alphabet to the end if necessary) 21 positions from V, the result would be the actual rightmost initial window letter. Going backwards on the ring of letters around the rotor 21 positions from V gives A, which is the actual rightmost initial window letter when the ciphertext was formed. Next, to find the ring setting of the rightmost rotor when the ciphertext was formed, we can use the fact that for the rightmost initial window letter X and ring setting 18 that resulted from the assumption that the last window letters prior to crib were ZZZ, or the actual rightmost initial window letter A and actual ring setting, the rotor core starting positions would have to be the same. For the rightmost initial window letter X and ring setting 18, since X is the 24th letter in the alphabet, the rotor core starting position would be $24 - 18 = 6$. Thus, because subtracting the rotor core starting position from the number of the initial window letter would give the ring setting, and the actual rightmost initial window letter A is the first letter in the alphabet, the actual ring setting would be 21, since $1 - 6 = -5 = 21 \bmod 26$. Clicking the **Find Actual Right Most...** button in the Maplet shows these results.

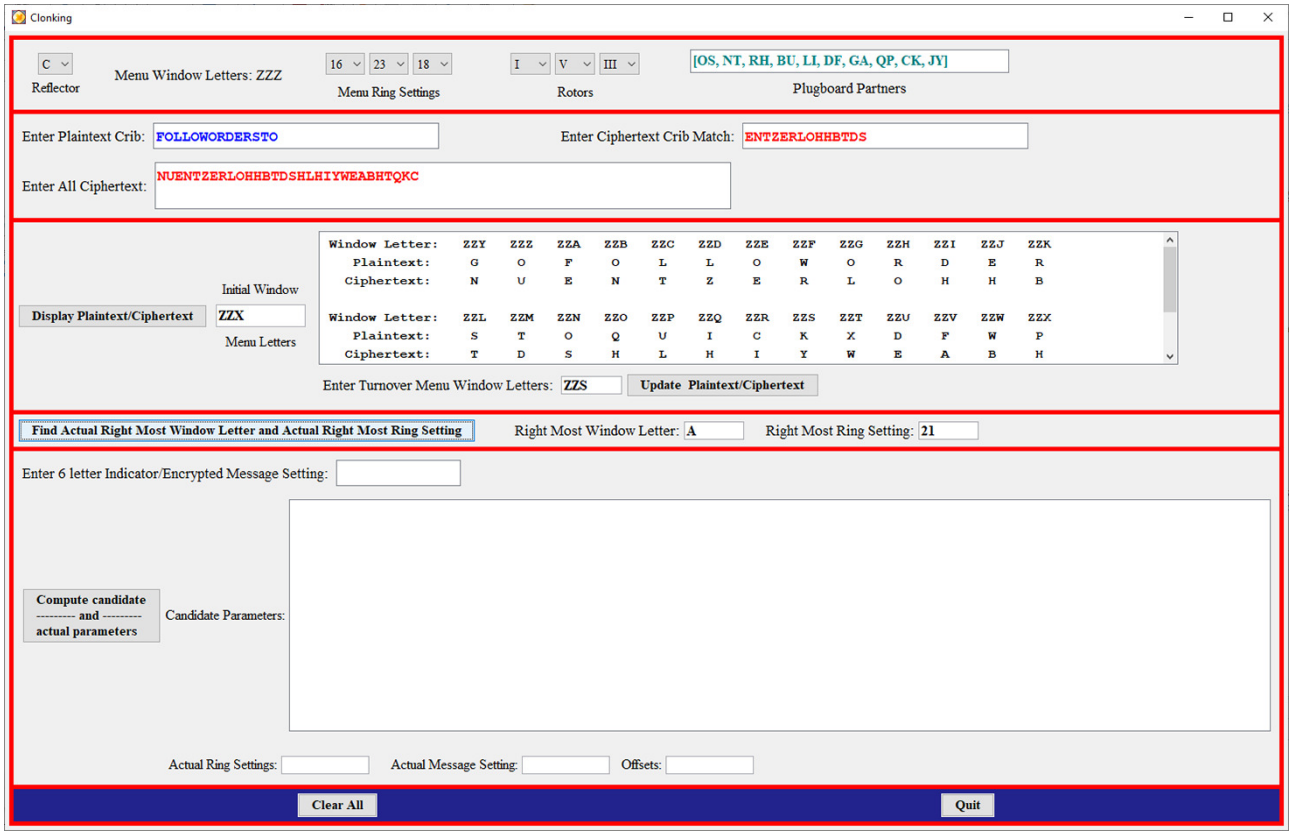

Clonking
C
Reflector
Menu Window Letters: ZZZ
16
23
18
Menu Ring Settings
I
V
III
Rotors
[OS, NT, RH, BU, LI, DF, GA, QP, CK, JY]
Plugboard Partners
Enter Plaintext Crib: FOLLOWORDERSTO
Enter Ciphertext Crib Match: ENTZERLOHHBTDS
Enter All Ciphertext: NUENTZERLOHHBTDSHLHIYWEABHTQKC
Window Letter: ZZY ZZZ ZZA ZZB ZZC ZZD ZZE ZZF ZZG ZZH ZZI ZZJ ZZK
Plaintext: G O F O L L O W O R D E R
Ciphertext: N U E N T Z E R L O H H B
Window Letter: ZZL ZZM ZZN ZZO ZZP ZZQ ZZR ZZS ZZT ZZU ZZV ZZW ZZX
Plaintext: S T O Q U I C K X D F W P
Ciphertext: T D S H L H I Y W E A B H
Initial Window
ZZX
Menu Letters
Display Plaintext/Ciphertext
Enter Turnover Menu Window Letters: ZZS
Update Plaintext/Ciphertext
Find Actual Right Most Window Letter and Actual Right Most Ring Setting
Right Most Window Letter: A
Right Most Ring Setting: 21
Enter 6 letter Indicator/Encrypted Message Setting:
Compute candidate -------- and -------- actual parameters
Candidate Parameters:
Actual Ring Settings:
Actual Message Setting:
Offsets:
Clear All
Quit

Figure 4: Finding an actual rightmost window letter and ring setting.

After finding the actual initial window letter and ring setting of the rightmost rotor, Allied codebreakers could then find the actual ring settings of the middle and leftmost rotors by exploiting how they knew Army and Air Force Enigma operators transmitted their message settings as described in Section 3. To demonstrate this, consider an Enigma configured with the correct reflector, rotor order, plugboard connections, and actual initial window letter and ring setting of the rightmost rotor, and with the rotors turned so that the indicator setting is showing through the windows. There would only

be $26 \cdot 26 = 676$ different possible combinations of ring settings of the middle and leftmost rotors. If the letters in the operator's encrypted message setting were decrypted assuming each of these 676 different possible combinations, only a relatively small number should cause the third letter to decrypt to the actual rightmost initial window letter. It would not be unreasonable to try all 676 different possible combinations to see which ones caused the third letter in the operator's encrypted message setting to decrypt to the actual rightmost initial window letter, and this is exactly what the codebreakers did. They actually did not even have to decrypt all of the letters in the operator's encrypted message setting for every trial. By rotating the rightmost rotor two positions forward to replicate the rotations that would occur when the first two letters were decrypted, they could initially test a combination by decrypting just the third letter, and then decrypt the first two letters only if the third letter decrypted to the actual rightmost initial window letter.

For example, suppose that the sequence `BOXFUW` consisting of the indicator setting and encrypted message setting was intercepted. For the sequence `BOXFUW`, the codebreakers could, with the rotors turned so that BOZ was showing in the windows, try all 676 different possible combinations of ring settings of the middle and leftmost rotors to see which ones caused `W` to decrypt as `A`. Then only for those that did, they could turn the rotors back so BOX was showing in the windows and decrypt `FU`.

The Allied codebreakers referred to the process that we just described as *clonking*. After clonking, from the combinations of ring settings of the middle and leftmost rotors that caused the third letter in the operator's encrypted message setting to decrypt to the actual rightmost initial window letter, they could identify the correct actual ring settings of the middle and leftmost rotors by checking to see which ones together with the resulting decrypted letters gave the same rotor core starting positions as the initial window letters and ring settings that had worked in the cryptanalysis process on the individual ciphertext.

As an demonstration of clonking, consider the example we began in Figure 4, in which we found that the actual initial window letter and ring setting of the rightmost rotor when the ciphertext was formed were A and 21, respectively. Clonking for the sequence `BOXFUW` would involve trying all 676 different possible combinations of ring settings of the middle and leftmost rotors to see which ones caused `FUW` to decrypt as `..A`, and then for each combination that did, finding the resulting rotor core starting positions. By entering BOXFUW and clicking the **Compute candidate...** button, the **Clonking** Maplet displays all ring settings of the middle and leftmost rotors that decrypt `FUW` as `..A`. The results are shown in Figure 5. By examining the output, we can see for example that the ring settings 3, 23, 21 cause `FUW` to decrypt as `ZFA`. Since ZFA are the alphabet letters in positions 26, 6, 1, the ring settings 3, 23, 21 with these initial window letters result in the rotor core starting positions 23, 9, 6, since $26-3=23$, $6-23=-17=9 \bmod 26$, and $1-21=-20=6 \bmod 26$. The initial window letters and ring settings that had worked in the cryptanalysis process on the ciphertext in this example were ZZX and 16, 23, 18, respectively. Since ZZX are the alphabet letters in positions 26, 26, 24, the ring settings 16, 23, 18 with these initial window letters result in the rotor core starting positions 10, 3, 6, since $26-16=10$, $26-23=3$, and $24-18=6$. Note that in the Maplet output, the ring settings 10, 15, 21 and decrypted letters `TRA` give these same rotor core starting positions. Thus, the actual initial window letters and ring settings when the ciphertext in this example was formed were TRA and 10, 15, 21. As a result, 10, 15, 21 must be the ring settings in the daily key for all Enigma ciphertexts formed on the same day as the ciphertext in this example. This result is verified in the Maplet. With the correct ring settings found for the daily key for the particular day, the Allied codebreakers could use the indicator setting transmitted by each Enigma operator, decrypt their message setting, and then decrypt the ciphertext message sent by the operator.

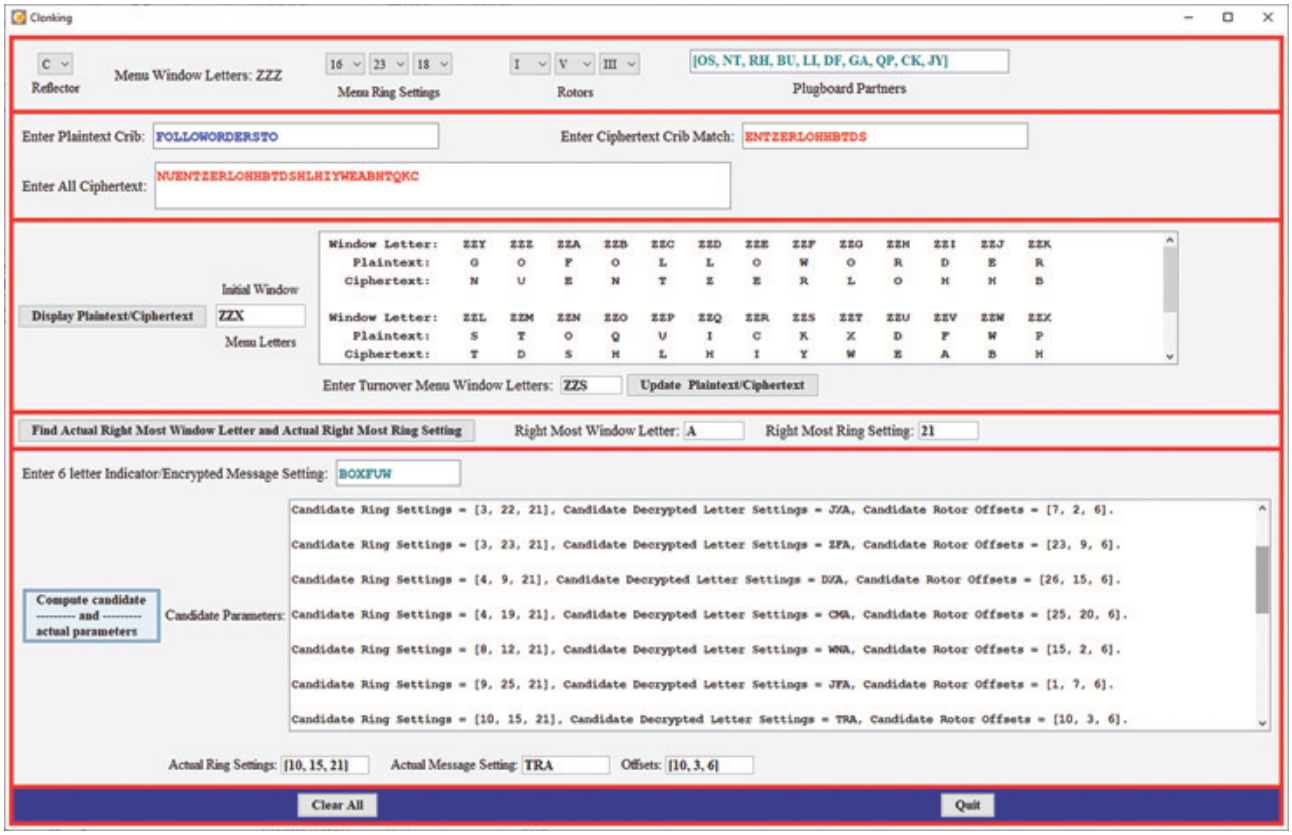

Clonking
C
Reflector
Menu Window Letters: ZZZ
16 23 18
Menu Ring Settings
I V III
Rotors
[OS, NT, RH, BU, LI, DF, GA, QP, CK, JV]
Plugboard Partners
Enter Plaintext Crib: FOLLOWORDERSTO
Enter Ciphertext Crib Match: ENTZERLOHHBTDS
Enter All Ciphertext: NUENTZERLOHHBTDSHLHIYWEABHTQKC
Window Letter: ZZY ZZZ ZZA ZZB ZZC ZZD ZZE ZZF ZZG ZZH ZZI ZZJ ZZK
Plaintext: G O F O L L O W O R D E R
Ciphertext: N U E N T Z E R L O H H B
Window Letter: ZZL ZZM ZZN ZZO ZZP ZZQ ZZR ZZS ZZT ZZU ZZV ZZW ZZX
Plaintext: S T O Q U I C K X D F W P
Ciphertext: T D S H L H Y W E A B H
Initial Window
Display Plaintext/Ciphertext
ZZX
Menu Letters
Enter Turnover Menu Window Letters: ZZS
Update Plaintext/Ciphertext
Find Actual Right Most Window Letter and Actual Right Most Ring Setting
Right Most Window Letter: A
Right Most Ring Setting: 21
Enter 6 letter Indicator/Encrypted Message Setting: BOXFUN
Compute candidate and actual parameters
Candidate Parameters:
Candidate Ring Settings = [3, 22, 21], Candidate Decrypted Letter Settings = JVA, Candidate Rotor Offsets = [7, 2, 6].
Candidate Ring Settings = [3, 23, 21], Candidate Decrypted Letter Settings = IFA, Candidate Rotor Offsets = [23, 9, 6].
Candidate Ring Settings = [4, 9, 21], Candidate Decrypted Letter Settings = DVA, Candidate Rotor Offsets = [26, 15, 6].
Candidate Ring Settings = [4, 19, 21], Candidate Decrypted Letter Settings = CMA, Candidate Rotor Offsets = [25, 20, 6].
Candidate Ring Settings = [8, 12, 21], Candidate Decrypted Letter Settings = WNA, Candidate Rotor Offsets = [15, 2, 6].
Candidate Ring Settings = [9, 25, 21], Candidate Decrypted Letter Settings = JFA, Candidate Rotor Offsets = [1, 7, 6].
Candidate Ring Settings = [10, 15, 21], Candidate Decrypted Letter Settings = TRA, Candidate Rotor Offsets = [10, 3, 6].
Actual Ring Settings: [10, 15, 21]
Actual Message Setting: TRA
Offsets: [10, 3, 6]
Clear All
Quit

Figure 5: Clonking to find actual message setting and ring settings.

5 Navy Encryption

At the beginning of World War II, the German Navy used the same three-rotor Enigma machine as the German Army and Air Force. However, in addition to the five rotors used by the Army and Air Force, Navy Enigma operators were equipped with three additional rotors, for a total of eight. Like Army and Air Force Enigma operators, Navy operators were issued a monthly setting sheet with instructions on how to configure their machine on any given day. Unlike their Army and Air Force counterparts though, Naval Enigma daily keys were issued in two parts. First, the *inner settings* specified the rotor order and ring settings that were to be used for pairs of days for the month covered by the setting sheet. Then, the *outer settings*, which were changed daily, specified the plugboard pairs and a collection of special positions specified by the rotor window letters known as the *Grundstellung* (in English, *initial position*).

In contrast to their counterparts in the German Army and Air Force, German Navy Enigma operators did not choose their own message settings. Instead, each operator selected a group of three letters called the *procedure indicator group* from a designated section of a printed book called the *K-book*. Then, after configuring their Enigma machine with the specified rotor order, ring settings, plugboard pairs, and window letters specified by the Grundstellung, the sending Enigma operator encrypted the three letters specified by the procedure indicator group. The three letters resulting from this encryption represented the message setting. The process was referred to by the Allied codebreakers as *Dolphin*.

Suppose, for example, that the daily key is the following.

Inner Settings:	Ring Settings:	16, 10, 22
	Rotor Order:	**VIII**, **IV**, **I**
Outer Settings:	Reflector:	B
	Plugboard Pairs:	O/S, N/T, R/H, B/U, L/I, D/F, G/A, Q/P, C/K, J/Y
	Grundstellung:	YES

To create the message setting, suppose the operator uses the K-book to look up procedure indicator ATS. The sending operator would first set up the machine using the inner and outer settings with the Grundstellung as the window letters, and use this setup to encrypt the procedure indicator. Figure 6 shows the result of using the **Encrypt/Decrypt Enigma Message** Maplet to do this, with output `SJK`.

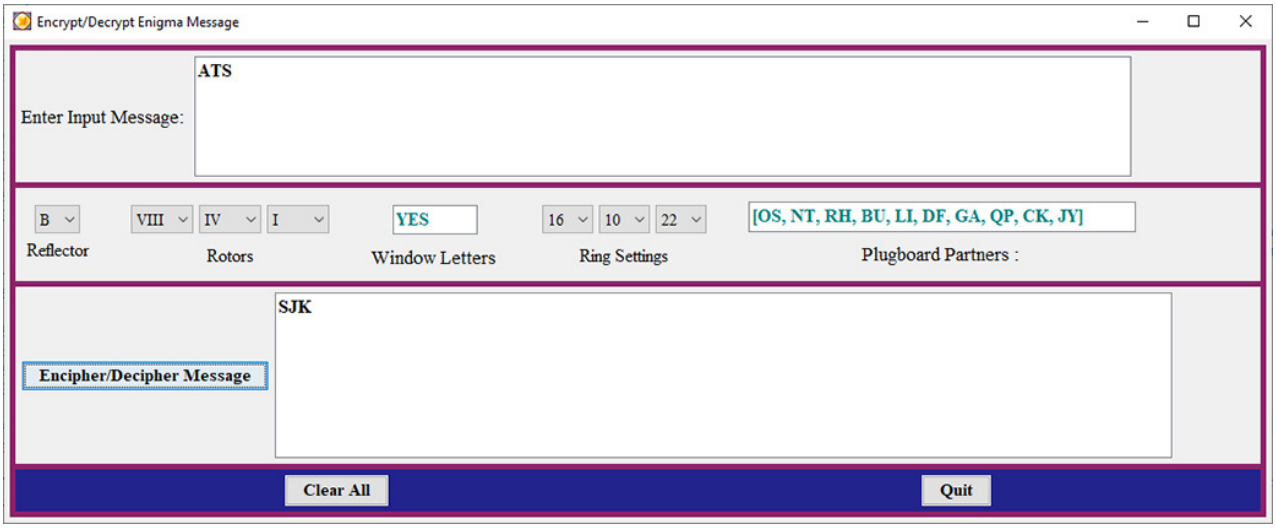


Figure 6: Encrypting a procedure indicator group.

Suppose now the sending operator wished to encrypt `TOMORROW IS BETTER`. They would then change the window letters in the machine to SJK, and use this setup to encrypt the message. Figure 7 shows the result of using the Maplet to do this, resulting in the ciphertext `CQCLWTIXNNLKFGGS`.

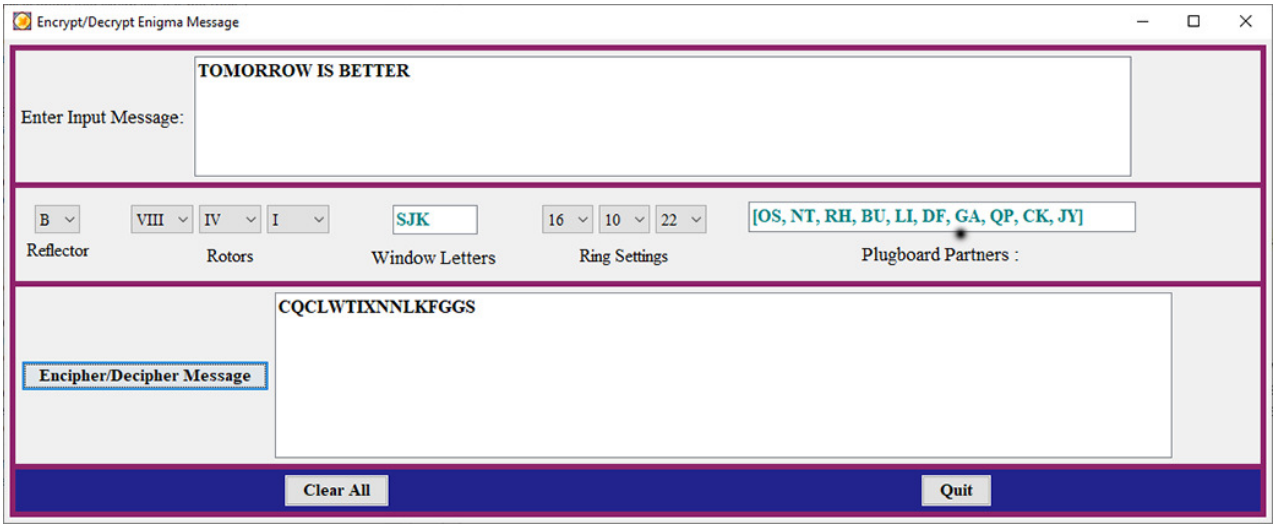
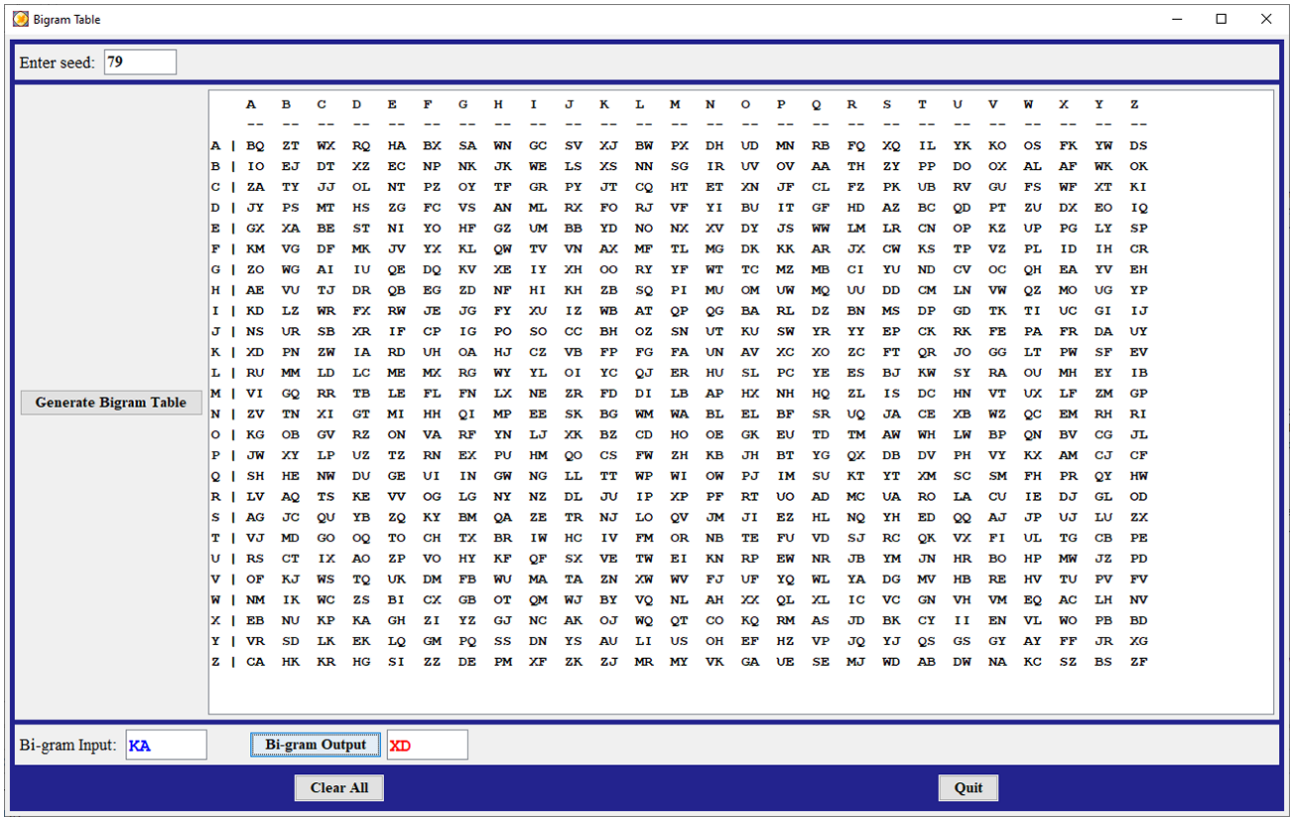


Figure 7: Encrypting a message.

German Navy Enigma sending operators used a different and more secure method than their counterparts in the Army and Air Force to transmit their message settings and ciphertexts to the receiving operators. The sending operator would encrypt the procedure indicator group by means of a *bi-gram* table. Bi-gram tables, which were arranged in 26×26 grids, were issued in a book to all Navy Enigma operators, with instructions for which bi-gram table was to be used on any given day. The **Bigram Table** Maplet, which was written by the authors, is designed to generate a random bi-gram table resulting from a given seed. For example, Figure 8 shows the bi-gram table generated with a random seed of 79.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	BQ	ZT	WX	RQ	HA	BX	SA	WN	GC	SV	XJ	BW	PX	DH	UD	MN	RB	FQ	XQ	IL	YK	KO	OS	FK	YW	DS
B	IO	EJ	DT	XZ	EC	NP	NK	JK	WE	LS	XS	NN	SG	IR	UV	OV	AA	TH	ZY	PP	DO	OX	AL	AF	WK	OK
C	ZA	TY	JJ	OL	NT	PZ	OY	TF	GR	PY	JT	CQ	HT	ET	XN	JF	CL	FZ	PK	UB	RV	GU	FS	WF	XT	KI
D	JY	PS	MT	HS	ZG	FC	VS	AN	ML	RX	FO	RJ	VF	YI	BU	IT	GF	HD	AZ	BC	QD	PT	ZU	DX	EO	IQ
E	GX	XA	BE	ST	NI	YO	HF	GZ	UM	BB	YD	NO	NX	XV	DY	JS	WW	LM	LR	CN	OP	KZ	UP	PG	LY	SP
F	KM	VG	DF	MK	JV	YX	KL	QW	TV	VN	AX	MF	TL	MG	DK	KK	AR	JX	CW	KS	TP	VZ	PL	ID	IH	CR
G	ZO	WG	AI	IU	QE	DQ	KV	XE	IY	XH	OO	RY	YF	WT	TC	MZ	MB	CI	YU	ND	CV	OC	QH	EA	YV	EH
H	AE	VU	TJ	DR	QB	EG	ZD	NF	HI	KH	ZB	SQ	PI	MU	OM	UW	MQ	UU	DD	CM	LN	VW	QZ	MO	UG	YP
I	KD	LZ	WR	FX	RW	JE	JG	FY	XU	IZ	WB	AT	QP	QG	BA	RL	DZ	BN	MS	DP	GD	TK	TI	UC	GI	IJ
J	NS	UR	SB	XR	IF	CP	IG	PO	SO	CC	BH	OZ	SN	UT	KU	SW	YR	YY	EP	CK	RK	FE	PA	FR	DA	UY
K	XD	PN	ZW	IA	RD	UH	OA	HJ	CZ	VB	FP	FG	FA	UN	AV	XC	XO	ZC	FT	QR	JO	GG	LT	PW	SF	EV
L	RU	MM	LD	LC	ME	MX	RG	WY	YL	OI	YC	QJ	ER	HU	SL	PC	YE	ES	BJ	KW	SY	RA	OU	MH	EY	IB
M	VI	GQ	RR	TB	LE	FL	FN	LX	NE	ZR	FD	DI	LB	AP	HX	NH	HQ	ZL	IS	DC	HN	VT	UX	LF	ZM	GP
N	ZV	TN	XI	GT	MI	HH	QI	MP	EE	SK	BG	WM	WA	BL	EL	BF	SR	UQ	JA	CE	XB	WZ	QC	EM	RH	RI
O	KG	OB	GV	RZ	ON	VA	RF	YN	LJ	XK	BZ	CD	HO	OE	GK	EU	TD	TM	AW	WH	LW	BP	QN	BV	CG	JL
P	JW	XY	LP	UZ	TZ	RN	EX	PU	HM	QO	CS	FW	ZH	KB	JH	BT	YG	QX	DB	DV	PH	VY	KX	AM	CJ	CF
Q	SH	HE	NW	DU	GE	UI	IN	GW	NG	LL	TT	WP	WI	OW	PJ	IM	SU	KT	YT	XM	SC	SM	FH	PR	QY	HW
R	LV	AQ	TS	KE	VV	OG	LG	NY	NZ	DL	JU	IP	XP	PF	RT	UO	AD	MC	UA	RO	LA	CU	IE	DJ	GL	OD
S	AG	JC	QU	YB	ZQ	KY	BM	QA	ZE	TR	NJ	LO	QV	JM	JI	EZ	HL	NQ	YH	ED	QQ	AJ	JP	UJ	LU	ZX
T	VJ	MD	GO	OQ	TO	CH	TX	BR	IW	HC	IV	FM	OR	NB	TE	FU	VD	SJ	RC	QK	VX	FI	UL	TG	CB	PE
U	RS	CT	IX	AO	ZP	VO	HY	KF	QF	SX	VE	TW	EI	KN	RP	EW	NR	JB	YM	JN	HR	BO	HP	MW	JZ	PD
V	OF	KJ	WS	TQ	UK	DM	FB	WU	MA	TA	ZN	XW	WV	FJ	UF	YQ	WL	YA	DG	MV	HB	RE	HV	TU	PV	FV
W	NM	IK	WC	ZS	BI	CX	GB	OT	QM	WJ	BY	VQ	NL	AH	XX	QL	XL	IC	VC	GN	VH	VM	EQ	AC	LH	NV
X	EB	NU	KP	KA	GH	ZI	YZ	GJ	NC	AK	OJ	WQ	QT	CO	KQ	RM	AS	JD	BK	CY	II	EN	VL	WO	PB	BD
Y	VR	SD	LK	EK	LQ	GM	PQ	SS	DN	YS	AU	LI	US	OH	EF	HZ	VP	JQ	YJ	QS	GS	GY	AY	FF	JR	XG
Z	CA	HK	KR	HG	SI	ZZ	DE	PM	XF	ZK	ZJ	MR	MY	VK	GA	UE	SE	MJ	WD	AB	DW	NA	KC	SZ	BS	ZF

Figure 8: Finding a bi-gram table.

Bi-gram tables were used to replace groups of two letters. For example, with a given bi-gram table, the replacement for the two-letter bi-gram KA would be the two letter sequence in the position in the table where the row labeled with the letter K intersected the column labeled with the letter A. For the table shown in Figure 8, this replacement for KA is XD. Note that bi-gram tables were symmetric in the sense that for the two-letter sequence XD, the position in the table where the row labeled X intersects the column labeled D is KA.

German Navy Enigma sending operators used bi-gram tables to encrypt their procedure indicator groups. To demonstrate how this worked, as a continuation of the example in this section, recall that the procedure indicator ATS was chosen from the K-book. The operator would first attach a random letter, say J, to the end of this procedure indicator to obtain ATSJ. From the K-book, the Enigma sending operator would choose another 3-letter sequence, say PRO, and attach a random letter, say K, to the beginning of it to obtain KPRO. These two four letter sequences would then be stacked to obtain the following.

K	P	R	O
A	T	S	J

With, for example, the bi-gram table shown in Figure 8, the operator would make the replacements KA $\rightarrow$ XD, PT $\rightarrow$ DV, RS $\rightarrow$ UA, and OJ $\rightarrow$ XK to obtain the following.

X	D	U	X
D	V	A	K

The Enigma sending operator would then transpose this array into the following.

X	D	D	V
U	A	X	K

Finally, the sending operator would string these rows together as `XDDVUAXK`, and transmit this to the receiving operator twice with the ciphertext `CQCLWTIXNNLKFGGS` generated in Figure 7 in between: `XDDVUAXK CQCLWTIXNNLKFGGS XDDVUAXK`.

To decrypt the message, the receiving operator would begin by reversing the process of the bi-gram table to recover the procedure indicator group ATS. Then, using the Grundstellung setting of YES and the rest of the machine settings that were given in the daily key, the receiving operator would encrypt ATS to recover the message setting of SJK. Finally, using this message setting, the receiving operator would be able to decrypt the ciphertext `CQCLWTIXNNLKFGGS` to recover the plaintext `TOMORROW IS BETTER`.

6 Navy Cryptanalysis

The fact that the German Navy had 336 possible rotor orders as compared to 60 for the German Army and Air Force alone vastly increased the amount of Bombe time needed to discover the initial plugboard pairs, reflector, rotor order, and ring settings needed to break Enigma messages. Some of this additional challenge was overcome through a technique developed by Alan Turing called *banburismus* which eliminated certain rotor orders. The construction of more bombes to test rotor orders also helped. Even when an Enigma daily key was found with the successful breaking of a single message though, the Navy's use of bi-gram tables to encrypt procedure indicators better prevented the direct recovery of the message settings used by the other Enigma operators on any particular day.

With these added difficulties, it became imperative that the Allies capture German monthly setting sheets. The first such capture occurred in 1941, and as the war progressed, other captures occurred. This enabled progress to be made, since it allowed the daily key to be known for a limited number of days. However, without knowledge of the bi-gram tables used by each operator, it was still difficult to recover the message setting used by each Enigma operator. This problem was initially resolved by constructing each day what was known as an *EINS catalog*.

The German word *eins*, which translates in English to *one*, was suspected to occur most often in German Navy plaintext messages. This suspicion was later verified since it was confirmed that approximately 90% of German Navy messages contained at least one occurrence of *eins*. Using the daily key provided by the captured setting sheets, a daily catalog of four letter groups obtained by encrypting `EINS` at all of the $26^3 = 17{,}576$ possible window letter position groups was constructed. The daily construction of the catalog was an arduous task, and was first carried out by hand. Later, a machine known as *the baby* was used, which recorded the entries on punch cards.

The **EINS Encryption Finder** Maplet, which was written by the authors, is designed to generate EINS catalogs. The Maplet takes daily key parameters as input, and returns after the **Generate EINS Table** button is clicked the EINS catalog resulting from these parameters. As an example, Figure 9 shows the catalog table that results from a daily key. The table displays all possible ways in which EINS could be encrypted with the given daily key followed by the window letters that generate each encryption. For example, the result AABC NIB in the upper left corner of the table indicates that EINS encrypts as AABC when the window letters are NIB.

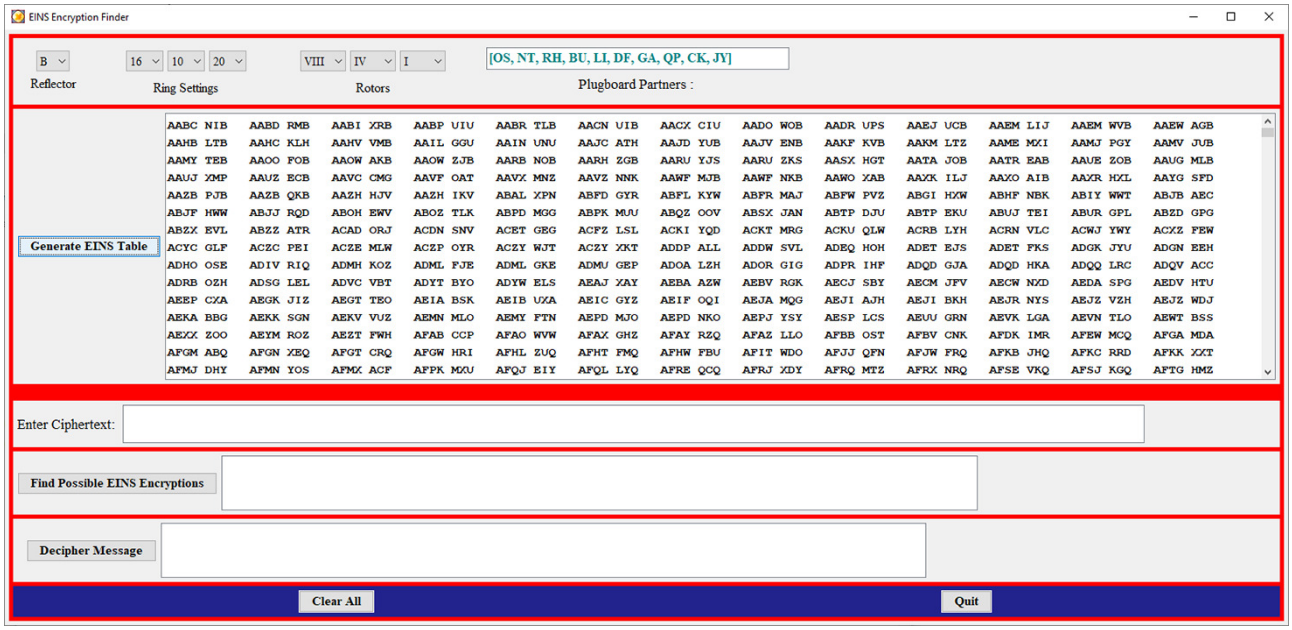

EINS Encryption Finder
B
Reflector
16
10
20
Ring Settings
VIII
IV
I
Rotors
[OS, NT, RH, BU, LI, DF, GA, QP, CK, JY]
Plugboard Partners :
AABC NIB AABD RMB AABI XRB AABP UIU AABR TLB AACN UIB AACX CIU AADO WOB AADR UPS AAEJ UCB AAEM LIJ AAEM WVB AAEW AGB
AAHB LTB AAHC KLH AAHV VMB AAIL GGU AAIN UNU AAJC ATH AAJD YUB AAJV ENB AAKF KVB AAKM LTZ AAME MXI AAMJ PGY AAMV JUB
AAMY TEB AAOO FOB AAOW AKB AAOW ZJB AARB NOB AARH ZGB AARU YJS AARU ZKS AASX HGT AATA JOB AATR EAB AAUE ZOB AAUG MLB
AAUJ XMP AAUZ ECB AAVC CMG AAVF OAT AAVX MNZ AAVZ NNK AAWF MJB AAWF NKB AAWO XAB AAXK ILJ AAXO AIB AAXR HXL AAYG SFD
AAZB PJB AAZB QKB AAZH HJV AAZH IKV ABAL XPN ABFD GYR ABFL KYW ABFR MAJ ABFW PVZ ABGI HXW ABHF NBK ABIY WWT ABJB AEC
ABJF HWW ABJJ RQD ABOH EWV ABOZ TLK ABPD MGG ABPK MUU ABQZ OOV ABSX JAN ABTP DJU ABTP EKU ABUJ TEI ABUR GPL ABZD GPG
ABZX EVL ABZZ ATR ACAD ORJ ACDN SNV ACET GEG ACFZ LSL ACKI YQD ACKT MRG ACKU QLW ACRB LYH ACRN VLC ACWJ YWY ACXZ FEW
ACYC GLF ACZC PEI ACZE MLW ACZP OYR ACZY WJT ACZY XKT ADDP ALL ADDW SVL ADEQ HOH ADET EJS ADET FKS ADGK JYU ADGN EEH
ADHO OSE ADIV RIQ ADMH KOZ ADML FJE ADML GKE ADMU GEP ADOA LZH ADOR GIG ADPR IHF ADQD GJA ADQD HKA ADQQ LRC ADQV ACC
ADRB OZH ADSG LEL ADVC VBT ADYT BYO ADYW ELS AEAJ XAY AEBA AZW AEBV RGK AECJ SBY AECM JFV AECW NXD AEDA SPG AEDV HTU
AEEP CXA AEGK JIZ AEGT TEO AEIA BSK AEIB UXA AEIC GYZ AEIF OQI AEJA MQG AEJI AJH AEJI BKH AEJR NYS AEJZ VZH AEJZ WDJ
AEKA BBG AEKK SGN AEKV VUZ AEMN MLO AEMY FTN AEPD MJO AEPD NKO AEPJ YSY AESP LCS AEUU GRN AEVK LGA AEVN TLO AEWT BSS
AEXZ ZOO AEYM ROZ AEZT FWH AFAB CCP AFAO WVW AFAX GHZ AFAY RZQ AFAZ LLO AFBB OST AFBV CNK AFDK IMR AFEW MCQ AFGA MDA
AFGM ABQ AFGN XEQ AFGT CRQ AFGW HRI AFHL ZUQ AFHT FMQ AFHW FBU AFIT WDO AFJJ QFN AFJW FRQ AFKB JHQ AFKC RRD AFKK XXT
AFMJ DHY AFMN YOS AFMX ACF AFPK MXU AFQJ EIY AFQL LYQ AFRE QCQ AFRJ XDY AFRQ MTZ AFRX NRQ AFSE VKQ AFSJ KGQ AFTG HMZ
Generate EINS Table
Enter Ciphertext:
Find Possible EINS Encryptions
Decipher Message
Clear All
Quit

Figure 9: Generating an EINS table.

Ciphertexts intercepted from German Navy Enigma operators were compared with the entries in the catalog to see if any of the four letter groups constructed by encrypting EINS also occurred in the ciphertexts. For a particular rotor position where a match occurred, the four characters in the ciphertext would decrypt to EINS. Once a match was located, further letters of the ciphertext would be decrypted to see if a legible plaintext was obtained.

As a demonstration of this process, and as a continuation of Figure 10, suppose the ciphertext BPDXHXWOPKPRCDKLWFGXVUTYL was intercepted. After entering this ciphertext in the **EINS Encryption Finder** Maplet, clicking the **Find Possible EINS Encryptions** button initiates a search for possible encryptions of EINS in the generated table. Figure 10 on the next page shows that there are three matching occurrences in the table, the locations in the ciphertext where each of these encryptions of **EINS** begins, and the window letters that give each one. Clicking the **Decipher Message** button then causes the full ciphertext to be decrypted with each possible set of window letters, and shows the full plaintext as the only legible option: ATCM OCCURS EINS TIME PER YEAR. Note that the Maplet also gives the actual window letters from the start of the encryption, FBI, which it finds by reverse-rotating the rotors the number of positions in the ciphertext prior to the encryption of EINS from the window letters that produced the encryption of EINS.

If approximately 200 message settings were recovered on any particular day, it became possible for Allied codebreakers to recover the bi-gram tables that were used. This involved a very laborious procedure though, and will not be discussed here.

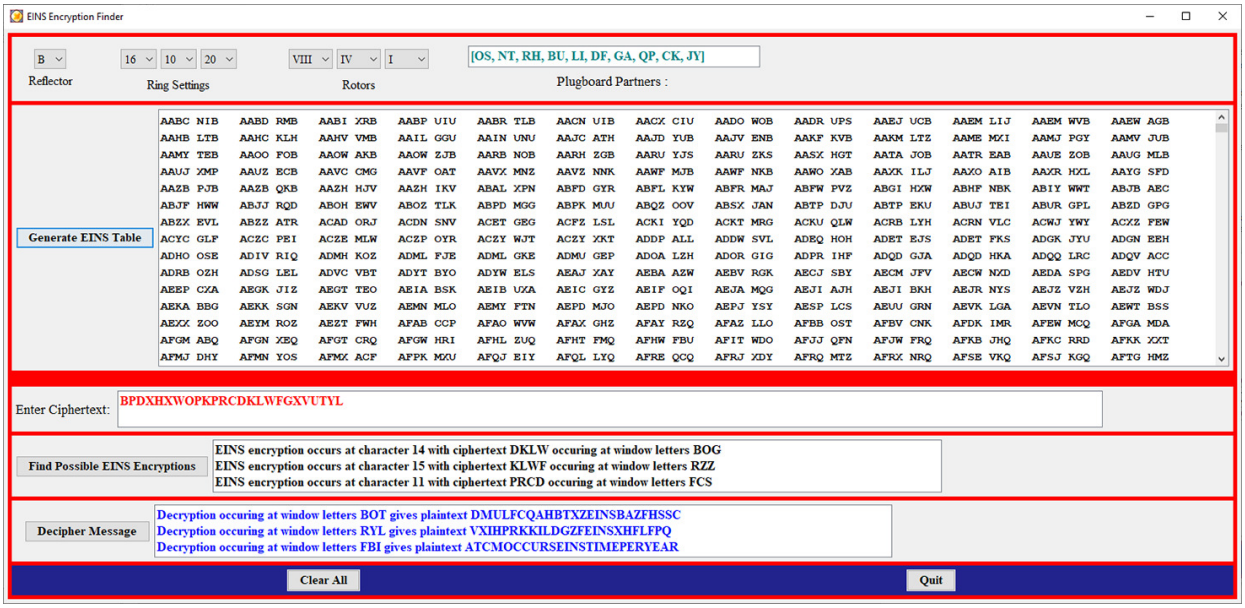

EINS Encryption Finder
B
Reflector
16
10
20
Ring Settings
VIII
IV
I
Rotors
[OS, NT, RH, BU, LI, DF, GA, QP, CK, JY]
Plugboard Partners :
Generate EINS Table
Enter Ciphertext:
BPDXHXWOPKPRCDKLWFGXVUTYL
Find Possible EINS Encryptions
EINS encryption occurs at character 14 with ciphertext DKLW occuring at window letters BOG
EINS encryption occurs at character 15 with ciphertext KLWF occuring at window letters RZZ
EINS encryption occurs at character 11 with ciphertext PRCD occuring at window letters FCS
Decipher Message
Decryption occuring at window letters BOT gives plaintext DMULFCQAHBTXZEINSBAZFHSSC
Decryption occuring at window letters RYL gives plaintext VXIHPRKKILDGZFEINSXHFLFPQ
Decryption occuring at window letters FBI gives plaintext ATCMOCCURSEINSTIMEPERYEAR
Clear All
Quit

Figure 10: Finding a message setting and decrypting a message.

7 Conclusion

In this paper, we extending our discussion that can be found in [2] to show how the German Army, Air Force, and Navy versions of the three-rotor Enigma machine could be cryptanalyzed and messages sent by all Enigma operators on any particular day read. Due to the more involved and secure method used by Navy operators to transmit their message settings, breaking Navy messages provided a much greater challenge than Army and Air Force messages.

Allied codebreaking efforts were complicated in February 1942 by the introduction of a four-rotor version of the machine by the German Navy. Called *Shark* by the Allies, keys for the four-rotor Enigma required further efforts to find, and required faster four-rotor bombes which were developed by the United States in 1943. A brief description of this process can be found in [1].

References

[1] Frank Carter, *Breaking Naval Enigma*, Bletchley Park Trust, Milton Keynes, UK, 2008.

[2] Richard Klima and Neil Sigmon, 2018. The Turing Bombe and Its Role in Breaking Enigma. Proceedings of the 22nd Asian Technology Conference in Mathematics (ATCM 2017). Available at: https://atcm.mathandtech.org/EP2017/invited/4202017_21528.pdf.

[3] Richard Klima and Neil Sigmon, 2021. Recognizing the Polish Efforts in Breaking Enigma. Proceedings of the 25th Asian Technology Conference in Mathematics (ATCM 2020). Available at: https://atcm.mathandtech.org/EP2020/invited/21799.pdf.

[4] Richard Klima and Neil Sigmon, *Cryptology, Classical and Modern, Second Edition*, Taylor & Francis, Boca Raton, FL, 2019.

[5] Neil Sigmon, 2024. Maplet Download Page for Cracking the Enigma Code: Beyond the Bombe. Available at: https://sites.radford.edu/~npsigmon/beyondthebombe/paper.html.